Argonne NATIONAL LABORATORY

Exascale Architecture Trends

Pete Beckman

Argonne National Laboratory

Northwestern University

U.S. DEPARTMENT OF ENERGY

HPC has been pretty successful…

Tianhe-2

Sequoia

K Computer

Old Wisdom:
Moore's Law = free exponential speedups!


THE WALL STREET JOURNAL.
WSJ
WEEKEND

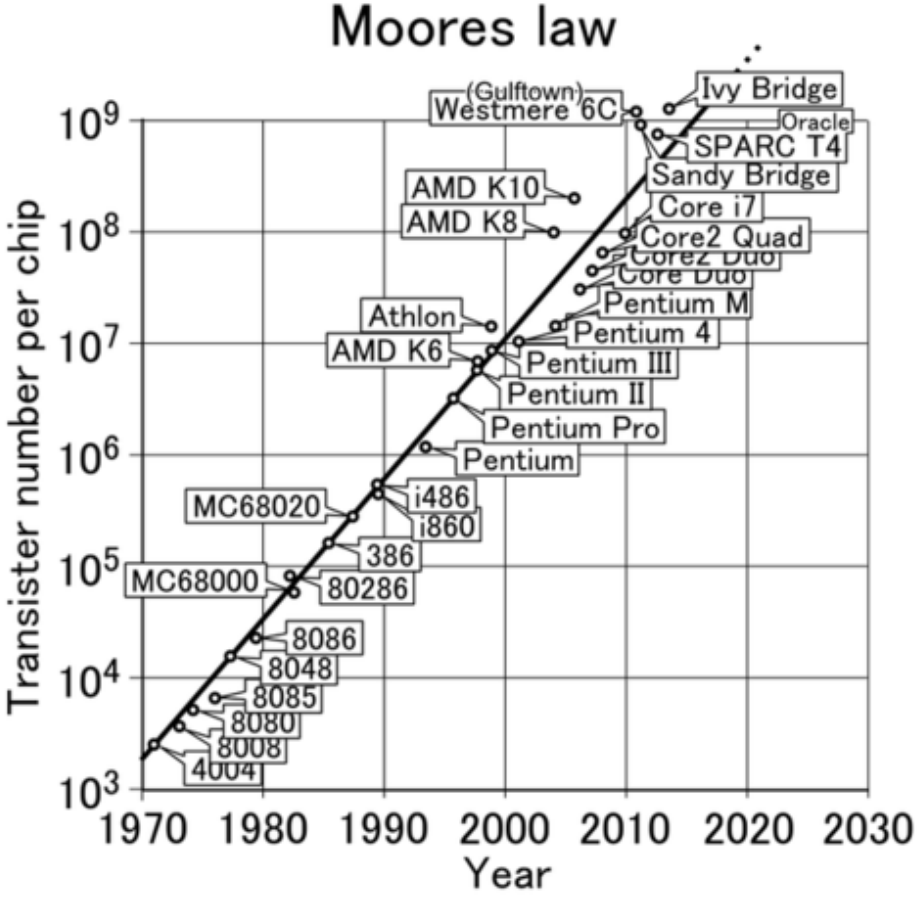
But the industry's costs keep rising, with new chip-fabrication plants costing as much as $10 billion. Cost pressures led International Business Machines Corp. last year to pay $1.5 billion to another company to take over its semiconductor operations.

Companies that can afford to keep pushing Moore's Law are finding it increasingly hard to keep up the pace. Intel's introduction of 14-nanometer technology was two quarters late be-


Turning 50, Tech Axiom Moore's Law Shows Age

By Don Clark

While companies say they likely can keep shrinking the size of silicon chips for another decade or so, that work is bringing diminishing financial returns. Some chip designers already are limiting their use of the newest technology to high-end products where performance is more important than cost.



Intel is about to save $1 billion by failing to keep up with a 50-year-old industry standard

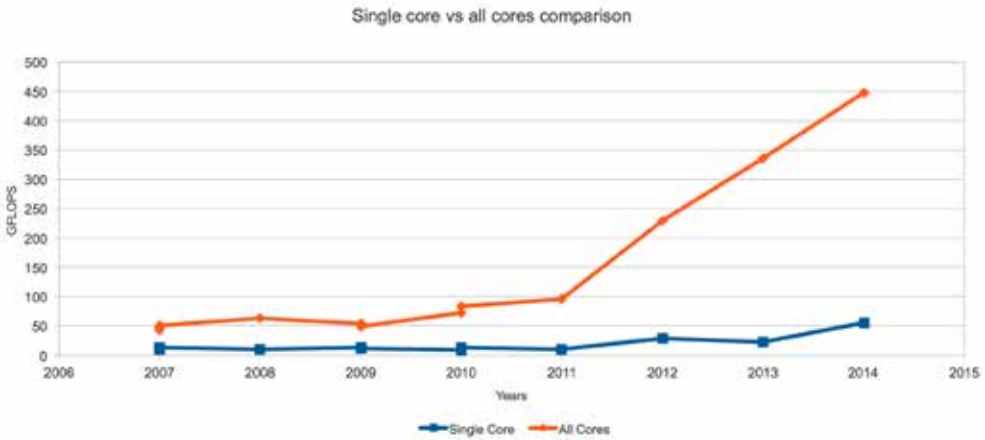
BUSINESS INSIDER

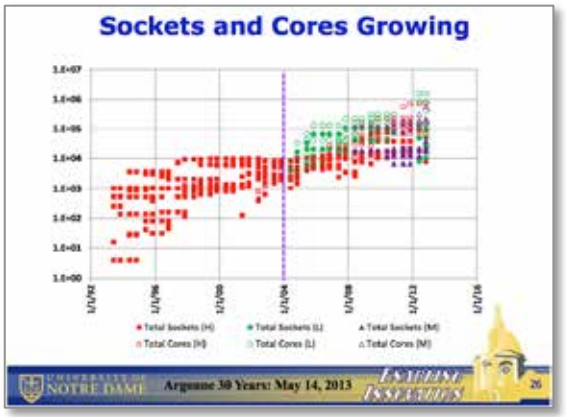
Intel: Moore's Law will continue through 7nm chips

Mark Hachman | @markhachman
Senior Editor, PCWorld
Feb 22, 2015 12:00 PM

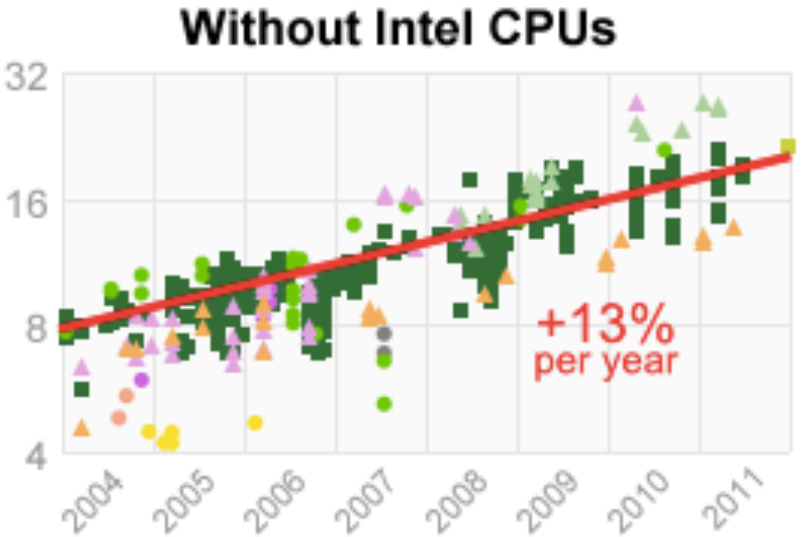
Reality: Computing improvements have slowed dramatically over the past Decade

Transistors you can buy for a fixed # of dollars in leading technology is no longer increasing!





*"No Moore?", Economist, Nov 2013.
Src: Linley Group



Single thread performance improvement is slow. (Specint)

"Herbert Stein's Law: "If something cannot go on forever, it will stop,"

*"Intel has done a little better over this period, Increasing at 21% per year.

Courtesy: Andrew Chien

Intel to Acquire FPGA-Specialist Altera for $16.7 Billion

20 Comments

+ Add A Comment

by Ryan Smith on June 1, 2015 6:05 PM EST

Posted in CPUs FPGA Intel Altera Xeon

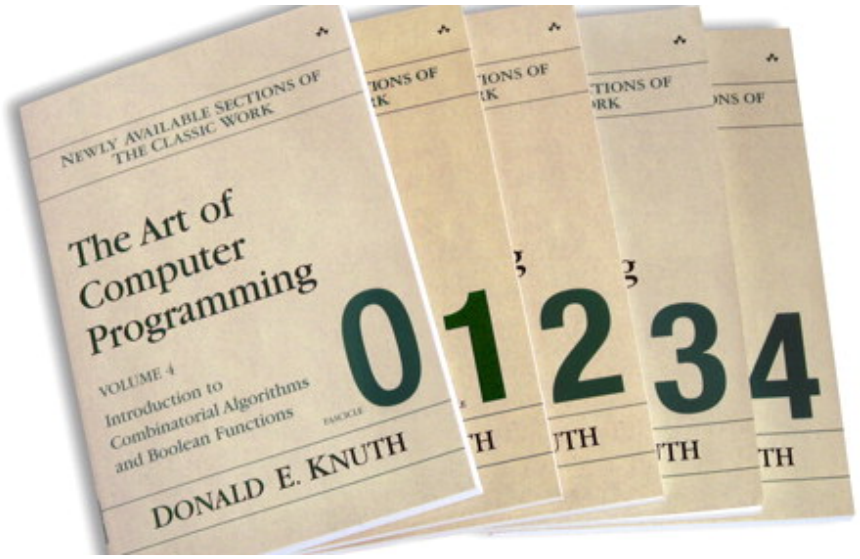
Today Intel has announced that they are buying Altera in an all-cash deal of $16.7 billion. The deal, having been rumored for a while now, will see Intel pick up Altera for their Field Programmable Grid Array (FPGA) experience, with Intel intending to both continue FPGA development and integrate FPGAs into some of their future products.

Old Wisdom:
Efficient Algorithms minimize operations

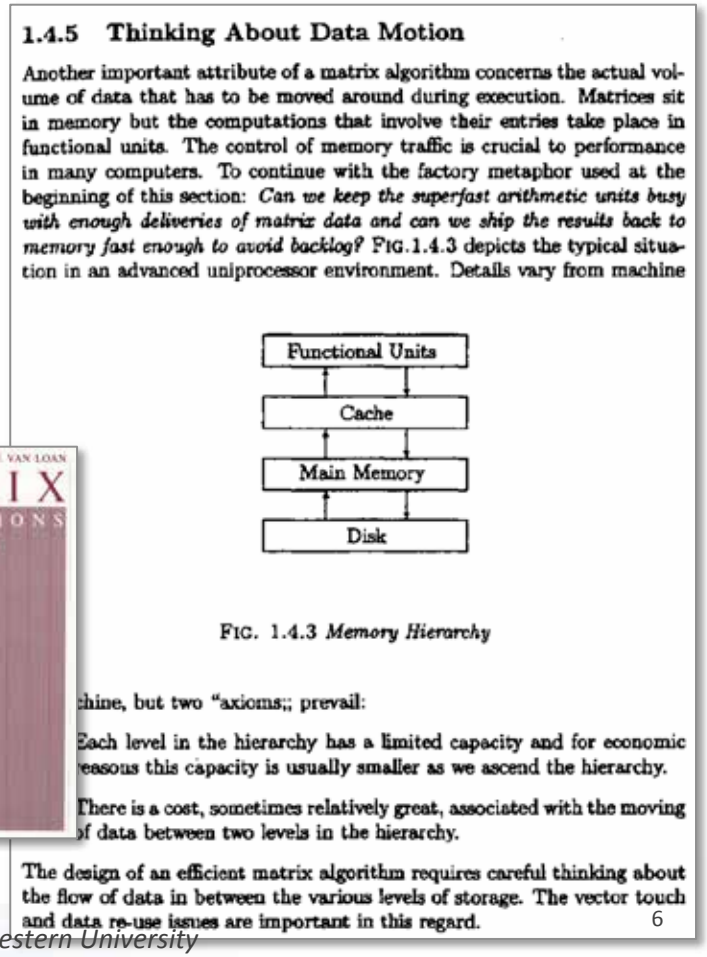
Classic Analysis of Algorithms: Ops = Time

Make algorithm quicker: minimize flops, compares

Ops: Best, Worst, Average, *Space*



1996



1.4.5 Thinking About Data Motion

Another important attribute of a matrix algorithm concerns the actual volume of data that has to be moved around during execution. Matrices sit in memory but the computations that involve their entries take place in functional units. The control of memory traffic is crucial to performance in many computers. To continue with the factory metaphor used at the beginning of this section: *Can we keep the superfast arithmetic units busy with enough deliveries of matrix data and can we ship the results back to memory fast enough to avoid backlog?* FIG.1.4.3 depicts the typical situation in an advanced uniprocessor environment. Details vary from machine

FIG. 1.4.3 *Memory Hierarchy*

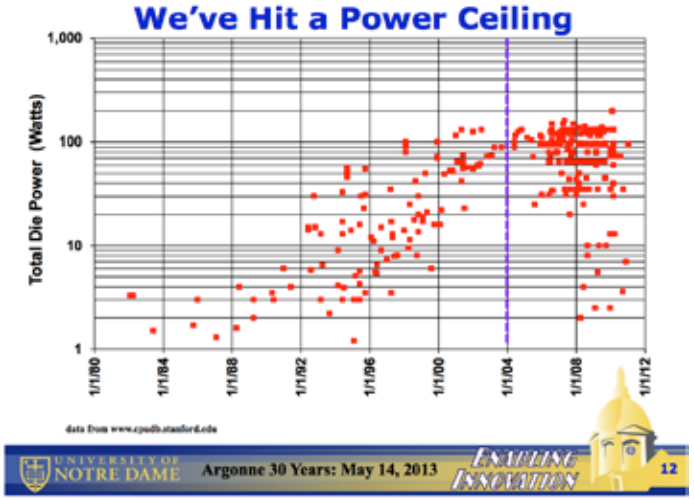
chine, but two "axioms;; prevail:

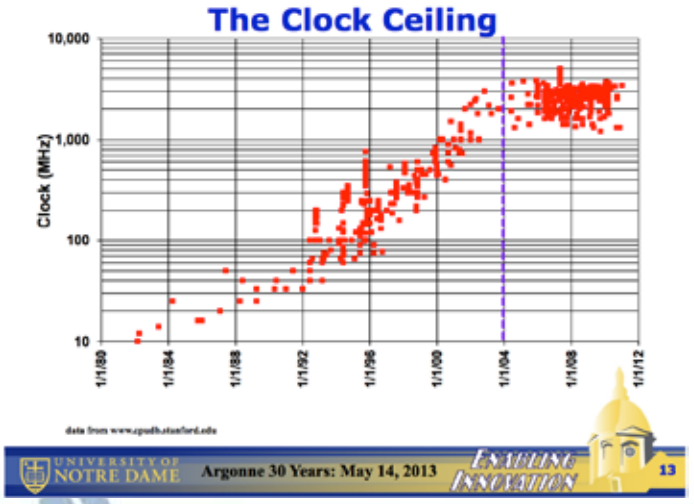
Each level in the hierarchy has a limited capacity and for economic reasons this capacity is usually smaller as we ascend the hierarchy.

There is a cost, sometimes relatively great, associated with the moving of data between two levels in the hierarchy.

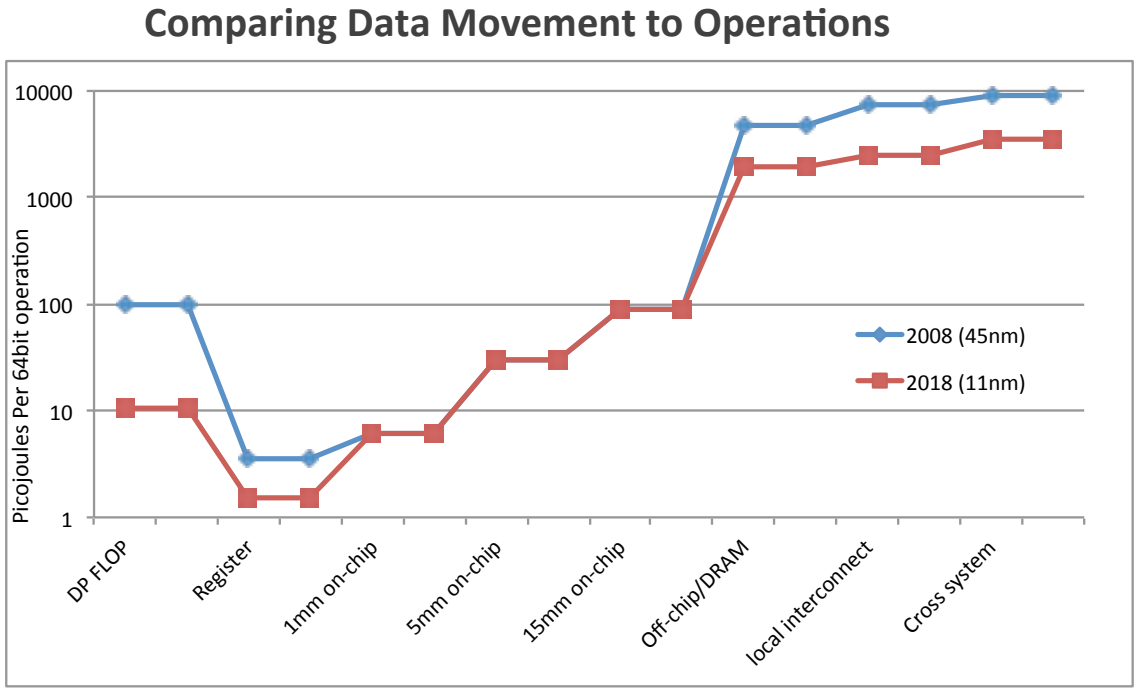
The design of an efficient matrix algorithm requires careful thinking about the flow of data in between the various levels of storage. The vector touch and data re-use issues are important in this regard.

Reality:
Efficient = optimize data movement (and power)





Courtesy: Peter Kogge



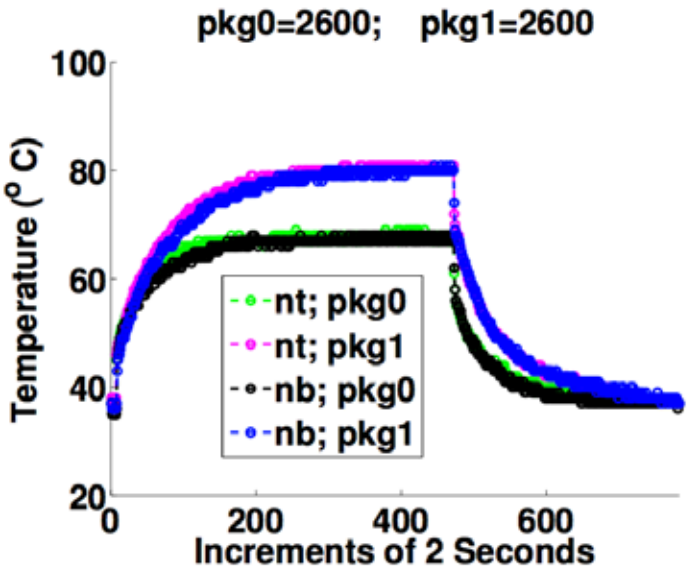
Courtesy: John Shalf

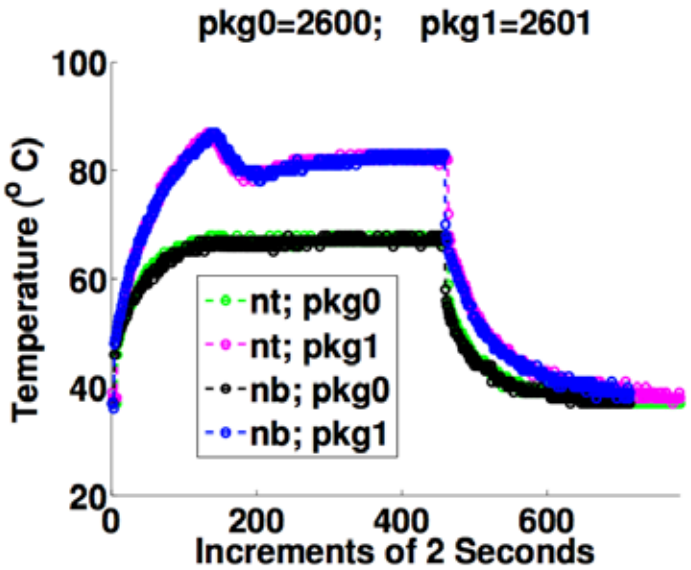
Pipelining, load/store, GPGU…

Old Wisdom:
Parallel Algorithms: Equal Work = Equal Time
(computers run at predictable speeds)

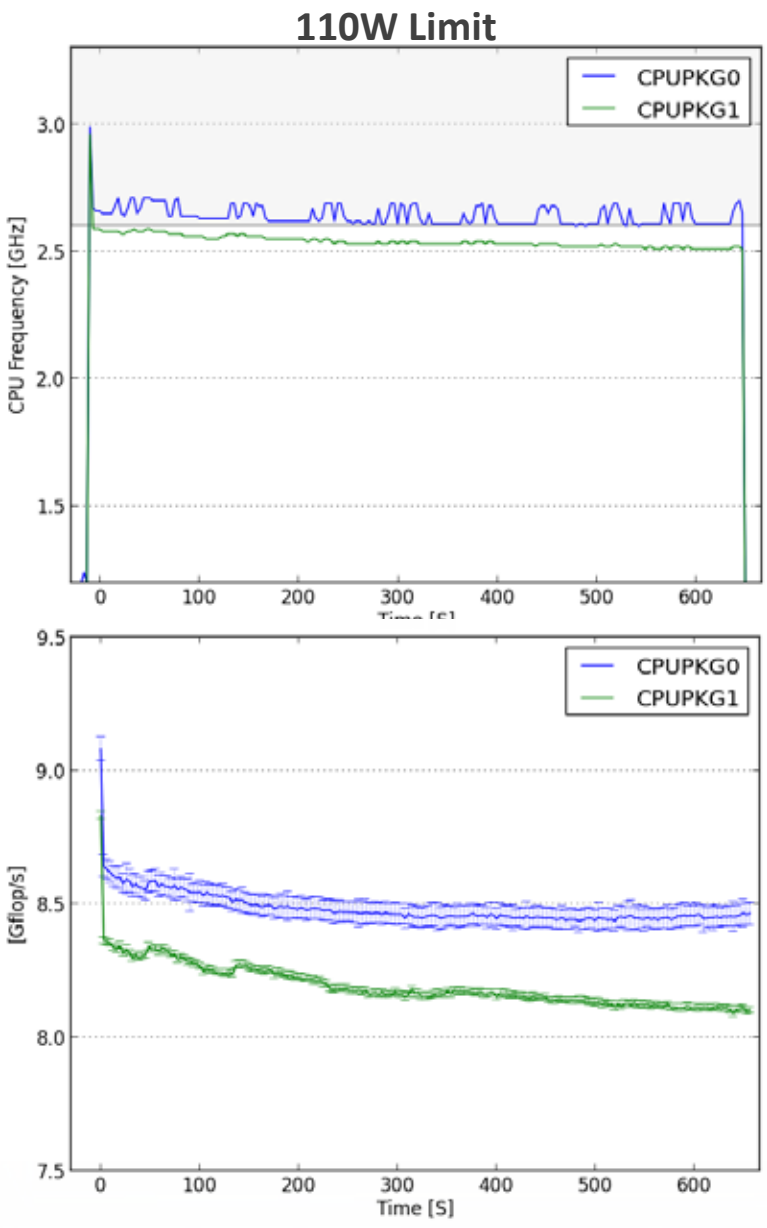
SPMD Code: Divide data into equal sized chucks across *p* processors

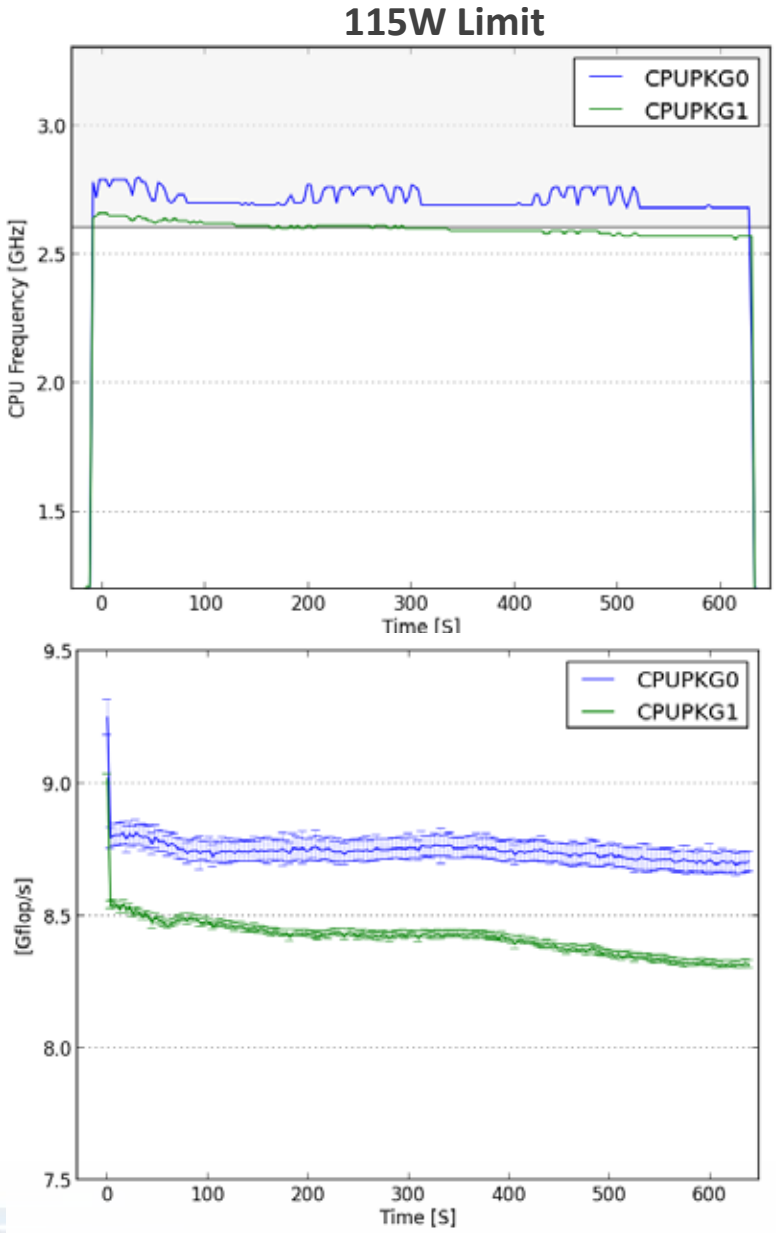
For all timesteps {
    exchange data with neighbors
    compute on local data
    barrier
}





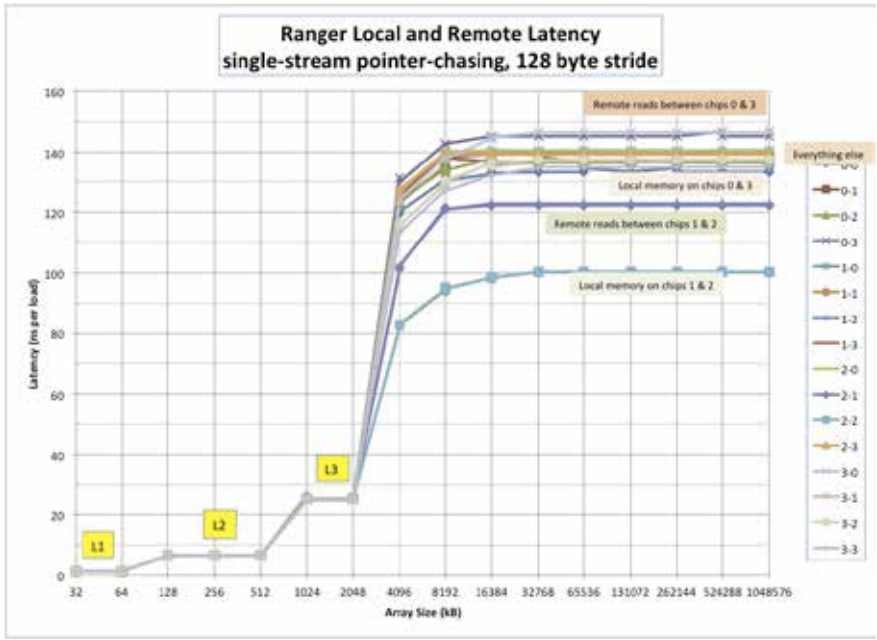
We live with dynamic now…





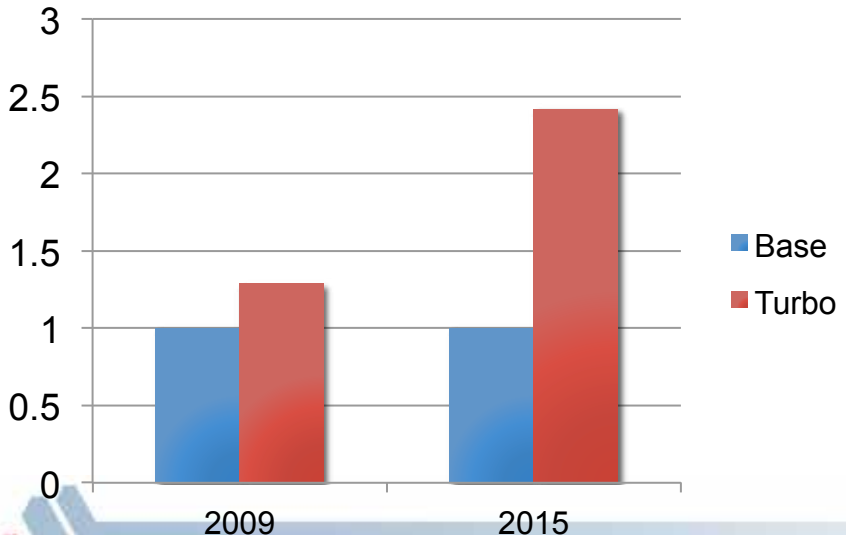
Reality: Performance is Highly Variable

Memory Hierarchy Depth (1-150-?)



Courtesy: McCalpin

Turbo Boost (1.2-2.5-?)



+ new Non-volatile memory (3,000 cycles)

+ old Non-volatile memory Flash (150,000 cycles)

Courtesy: Andrew Chien

The New Exascale Reality

- ~~Computing rapidly gets faster and cheaper for free~~
  - Rapid exponential improvement is over, slow improvement will continue for awhile... Parallelism explodes, SQUEEEEZE!
- ~~Efficient programs minimize operations~~
  - More operations can better, optimize for locality, data movement, power
- ~~Computers run at fixed, predictable speed~~
  - Increasing dynamic and flexible, complication and advantage

What Prevents Scalability?

(in the large and in the small)

- **Insufficient parallelism**
  - As the problem scales, more parallelism must be found
- **Insufficient latency hiding**
  - As the problem scales, more latency must be hidden
- **Insufficient resources** (Memory, BW, Flops)
  - As the problem scales, so must the resources needed

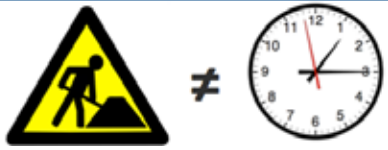
What Prevents Scalability?

(in the large and in the small)

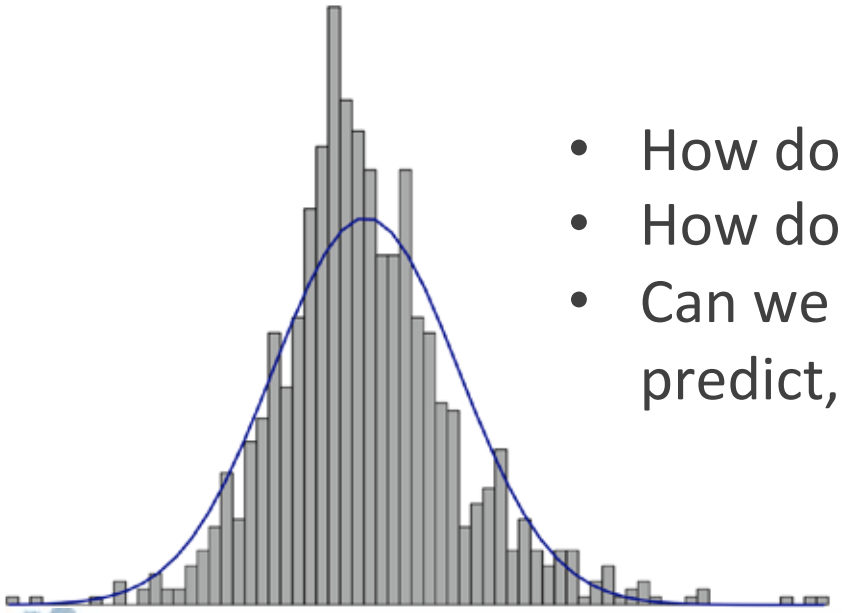
- **Insufficient parallelism**
  - As the problem scales, more parallelism must be found
- **Insufficient latency hiding**
  - As the problem scales, more latency must be hidden
- **Insufficient resources** (Memory, BW, Flops)
  - As the problem scales, so must the resources needed

As we scale machine, system becomes more dynamic
As we squeeze power, system becomes more dynamic
As we address resilience, system becomes more dynamic
As we share networks, system becomes more dynamic

Our Hardware is Dynamic, Adaptive Today!
(the future is even more dynamic)



- **Bulk Synchronous is our scaling problem**
  - **≠MPI (library that moves data with put/get or send/recv)**
  - **We must focus on dynamic behavior**
- **"OS Noise" and "jitter" is a legacy distraction**
  - **OS & Runtime must be VERY active…**
- **Load balancing is necessary, but not sufficient…**

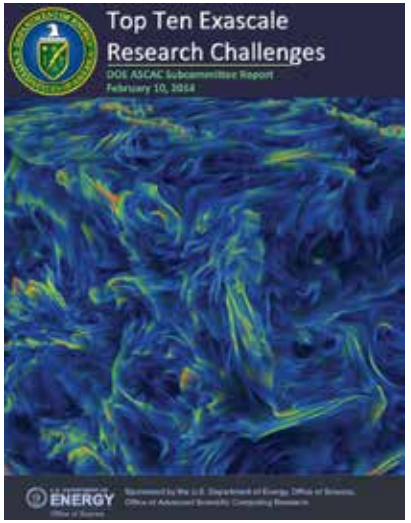


- How do we design software in this new era?
- How do we build latency tolerant algs?
- Can we create tools that measure, learn, predict, and then improve performance?

But yet, We Pretend our World is Not Dynamic

- **Trinity/NERSC-8:** ?

"The system shall provide correct and consistent runtimes. An application's runtime (i.e. wall clock time) shall not change by more than 3% from run-to-run in dedicated mode and 5% in production mode."



ASCAC Top 10 Research Challenges for Exascale

- "[…] power management [..] through dynamic adjustment of system balance to fit within a fixed power budget"
- […] Enabling […] dynamic optimizations […] (power, performance, and reliability) will be crucial to scientific productivity. "
- " […] Next-generation runtime systems are under development that support different mixes of several classes of dynamic adaptive functionality. "

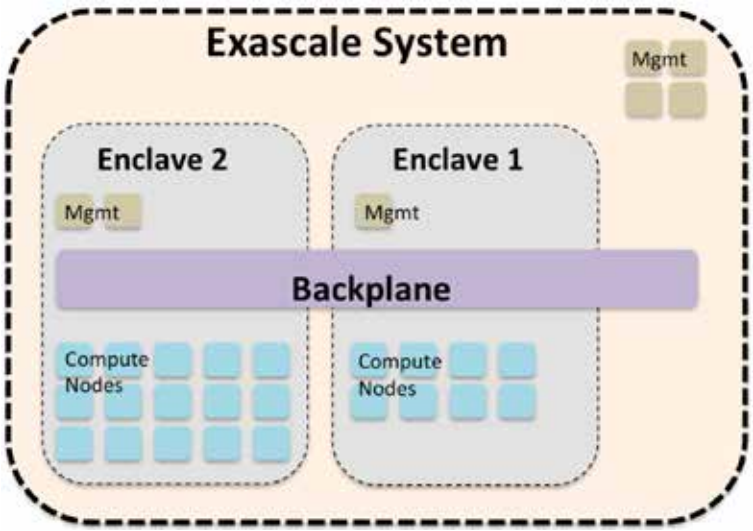
"dynamic" mentioned 43 times in 86 pg report

Lessons for the Future

- **Code should be as static as possible, but no more so**
- 1) Prepare: Create flexibility via over-decomposition, clear expression of dependencies
- 2) Take small steps to becoming more pliable…. statically
  - (static) mapping of resource (slow/fast; heat)
  - (static) load balancing (periodic repartitioning)
  - (static) dependency graph tiling of stencils to match communication
- 3) Find ***goal-oriented optimization***
  - Dynamic lightweight work-sharing
  - Dynamic power management
  - Dynamic data movement across hierarchy

Code should not consider dynamic a performance error (e.g. NERSC)

Exascale Operating System

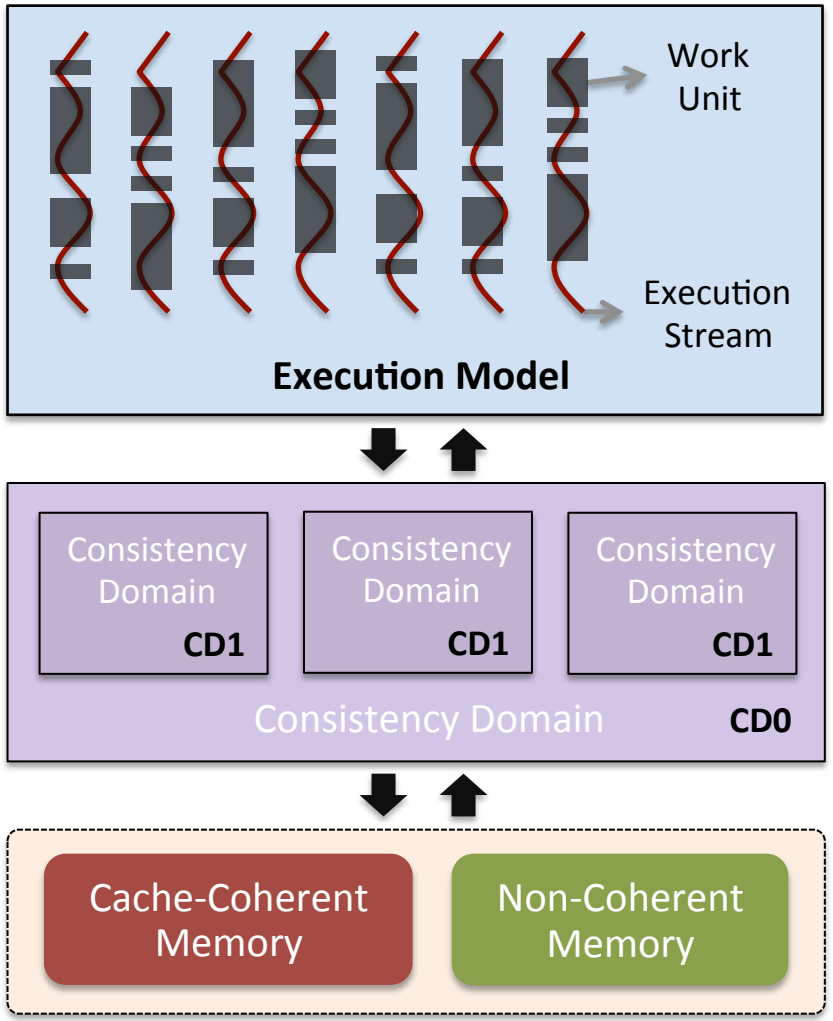


New abstractions & implementations

ANL: Pete Beckman, Marc Snir, Pavan Balaji, Rinku Gupta, Kamil Iskra,
Franck Cappello, Rajeev Thakur, Kazutomo Yoshii
LLNL: Maya Gokhale, Edgar Leon, Barry Rountree, Martin Schulz, Brian Van Essen
PNNL: Sriram Krishnamoorthy, Roberto Gioiosa
UC: Henry Hoffmann
UIUC: Laxmikant Kale, Eric Bohm, Ramprasad Venkataraman
UO: Allen Malony, Sameer Shende, Kevin Huck
UTK: Jack Dongarra, George Bosilca, Thomas Herault

Argobots

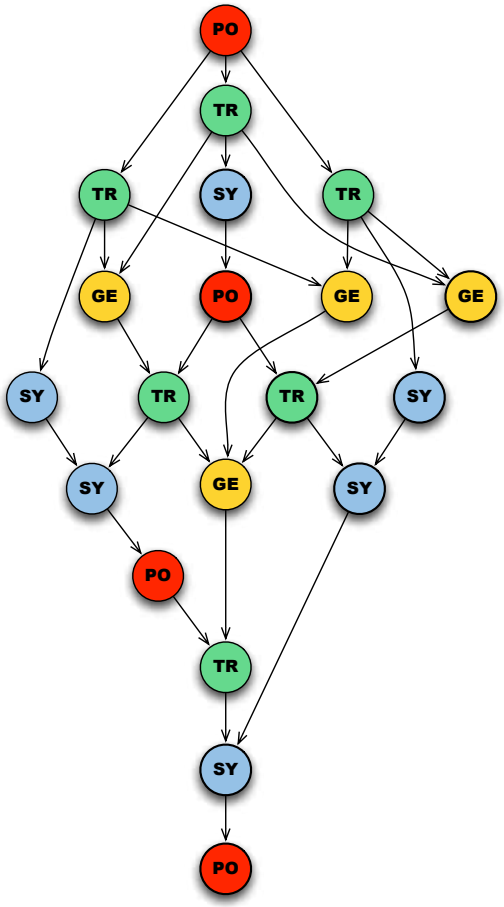
- **Lightweight Low-level Threading/ Tasking Framework**
- Separation of abstraction and mapping to implementation
- Massive parallelism
  - **Exec. Streams** guarantee progress
  - **Work Units** execute to completion
- Clearly defined memory semantics
  - Consistency domains
    - Provide Eventual Consistency
  - Software can manage consistency
  - Work Units can access any consistency domain
  - Support explicit memory placement and movement
- Put/Get/Send/Recv requires library call in OSR, but could be transparent at application level
- Exploring fault model and atomics



Threading-aware Task Schedulers

Task Schedulers+Argobots

- Task scheduling built on tasklets and user-level threads in argobots
- Focus on two classes of task graphs
  - Fork-join computations
  - Compact DAG representations
- Exploit the scheduling characteristics of argobots
  - Control over mapping threads to cores
  - Control over scheduling
  - Split-phase communication and task scheduling
- Initial Implementation
  - Argobots-optimized Cilk scheduler
  - Parallel Task Graph Engine (PTGE)

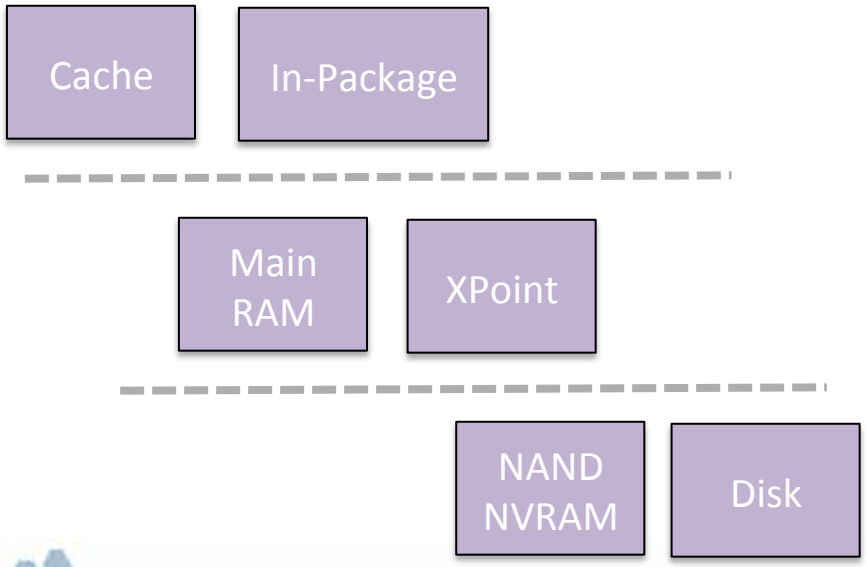


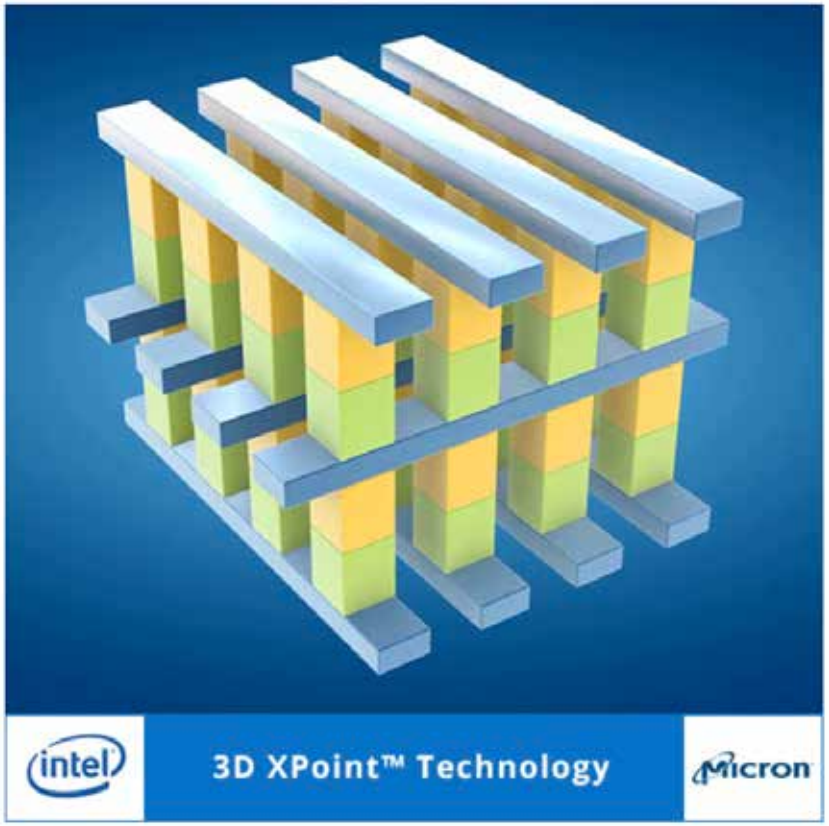
THE WALL STREET JOURNAL.

Intel, Micron Claim Chip Breakthrough

Companies say new memory chips are up to 1,000 times faster than NAND chips

Our Future is **Memory Hierarchy (adding dynamic behavior)**





Conclusions: The Times They are A-Changin'

- Embrace DYNAMIC!
  - Work ≠ Time
- Optimize algorithms for data movement
- Imagine multiple memory allocators
  - Manual data movement
- Learn to love runtime systems
- Explore adaptive, learning, predictive software stacks that takes humans out of the loop…
  - Sorry humans, you are too slow.
  - Reject human tuning papers…

Questions?